Modelos Probabilistas Aplicados

Johanna Bolaños Zúñiga

Matricula: 1883900

Tarea 11

## 1. Aplicación de la convolución

La convolución es un operador matemático que transforma dos funciones en una tercera función que representa la magnitud en la que se superponen una función sobre una versión trasladada e invertida de la otra función. De acuerdo a Kim [3], la convolución se puede describir como una función que es la integral (cuando las variables son continuas) o la suma de funciones (cuando las variables son discretas) de dos variables aleatorias independientes, y mide la cantidad de superposición cuando una función se desplaza sobre la otra. En las definiciones matemáticas la convolución, tanto discreta como continua, está indicada por el operador $*$.

En el procesamiento de señales digitales (DSP por sus siglas en inglés de *digital signal processing*), donde una de sus aplicaciones prácticas son el tratamiento de las imágenes, en la que la convolución juega un papel importante, ya que es una técnica de procesamiento de imágenes común que cambia las intensidades de un píxel para reflejar las intensidades de los píxeles circundantes. Además, se puede obtener efectos de imagen populares como desenfoque, nitidez y detección de bordes [4]. Un detector de bordes puede procesar y extraer características relevantes en un conjunto de imágenes antes de que se introduzcan en un algoritmo de reconocimiento de patrones, lo que puede dar como resultado un rendimiento superior, por ejemplo, que un automóvil sin conductor frena en una señal de alto. La mejora de imágenes puede resultar especialmente útil cuando se trata de imágenes científicas.

En el procesamiento de imágenes, muchas operaciones de filtro se aplican a una imagen realizando una operación especial llamada convolución con una matriz que recibe el nombre de kernel, los cuales son típicamente matrices cuadradas que van desde $2 \times 2$, hasta de $5 \times 5$, siendo la comúnmente utilizada la de $3 \times 3$. Los valores almacenados en el kernel se relacionan directamente con los resultados de la aplicación

del filtro a la imagen y son los que determinan cómo transformar los píxeles de la imagen original en los píxeles de la imagen procesada [4].

Dado que las imágenes también se pueden considerar como cuadrículas bidimensionales de números, la aplicación de un kernel a una imagen se puede visualizar como una pequeña cuadrícula (el kernel) que se mueve a través de una cuadrícula sustancialmente más grande (la imagen), donde la convolución del kernel hace que el valor de cada píxel se recalcula utilizando la suma de vecindad ponderada definida en la matriz del kernel. La representación de los pasos de esta convolución se puede observar en la figura 1.

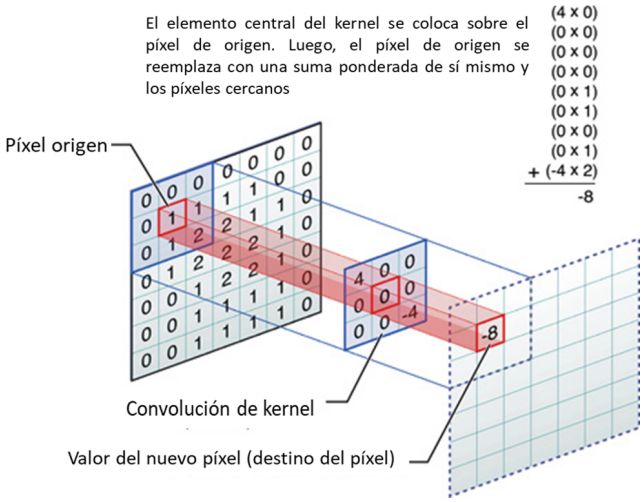


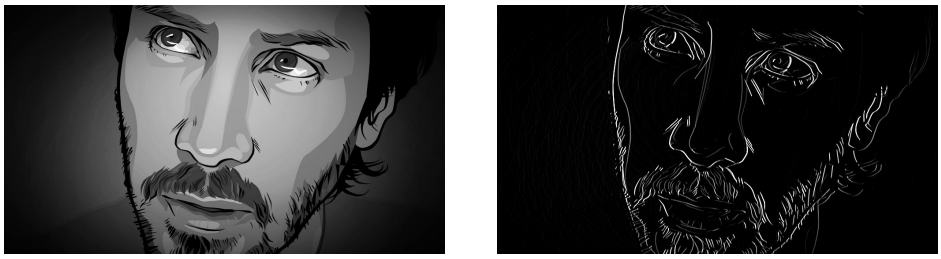

Figura 1: Ejemplo de convolución del kernel, tomado de la información de la referencia [4]

En la investigación desarrollada por Kim [3], realizan el tratamiento de imágenes a blanco y negro (escala de grises) mediante la convolución del kernel *Sobel*, el cual permite detectar los bordes verticales y horizontales en una imagen, obteniendo así el relieve de la imagen. En el programa R, mediante la función `magik()`, se pueden realizar el procesamiento a las imágenes y con la función `image_convolve()` se pueden realizar la convoluciones de kernel. En la figura 2 se muestra el resultado de aplicar la convolución con el kernel *Sobel* a una imagen. En la documentación de Thyssen [7] se encuentran más ejemplos de `convolve` con varios kernels disponibles. Además de aplicaciones como el relieve de la imagen, se pueden utilizar filtros para eliminar señales e imágenes (ruido). Diferentes filtros pueden lograr este propósito y el óptimo a menudo, depende de los requisitos particulares de la aplicación que se requiera.

De igual manera, en el trabajo de Kim [3], presentan las expresiones matemáticas para la convolución con imágenes, determinando que, para el caso de las señales digitales, si $An$ y $Bn$ son funciones con respecto a un valor $n$ entero, entonces la convolución en un sistema unidimensional, se representa mediante la ecuación 1:

$$A[n_a] * B[n_b] = C[n_c] = \sum_{\tau=0}^{n_a-1} A[\tau] \times B[n_c - \tau], \tag{1}$$

donde $0 \leq n_c < n_a + n_b - 1$ y $\tau$ es la variable sobre la cual se realiza el desplazamiento.



(a) Imagen antes de la convolución. Fuente: *A Scanner Darkly (2006).*

(b) Imagen después de la convolución con el kernel *Sobel*

Figura 2: Ejemplo de imagen aplicando convolución de kernel

Sin embargo, debido a que las imágenes son bidimensionales, es neceario la extensión a 2-D de la ecuación 1 para realizar la convolución con imágenes, la cuál es mostrada en la ecuación 2:

$$A[i_a,\, j_a] * B[i_b,\, j_b] = C[i_c,\, j_c] = \sum_{\tau_1=0}^{i_a-1} \sum_{\tau_2=0}^{j_a-1} A[\tau_1,\, \tau_2] \times B[i-\tau_1,\, j-\tau_2]. \tag{2}$$

En conclusión, gran parte de la tecnología que existe hoy en día no sería posible sin un medio para extraer información y manipular señales digitales, donde uno de los métodos para procesar imágenes digitales, llamado filtrado, se puede utilizar para reducir la información de la señal no deseada (ruido) o extraer información como los bordes de la imagen, lo cual se logra mediante un enfoque matemático utilizando la convolución y la matriz del kernel.

El código empleado para el tratamiento de las imágenes en el software R versión 4.0.2 [6] se encuentra en el repositorio GitHub [1].

## 2. Aplicación de la prueba de chi cuadrada ($\chi^2$)

Chi cuadrada ($\chi^2$) es la distribución de la suma de variables aleatorias cuadradas. Se utiliza la distribución $\chi^2$ para examinar si un conjunto de datos **difiere** de forma estadísticamente significativa de lo esperado, se conoce también como una prueba de calidad de ajuste y se requiere conocer la distribución que se espera ver y las frecuencias observadas de cada valor posible [5]. Se utiliza la ecuación 3 para determinar el valor de $\chi^2$, donde $k-1$ son grados de libertad:

$$\sum_{k} \frac{(\text{esperada-observada})^2}{\text{esperada}} \sim \chi^2(|k|-1). \tag{3}$$

En el tema de tesis que actualmente se desarrolla, se utiliza un GRASP (abreviatura de *greedy randomized adaptive search procedures*) reactivo para hallar la solución al problema. Este algoritmo depende de

Cuadro 1: Tabla de contingencia para la prueba $\chi^2$

| **Alfa** | **Frecuencia** | | **Diferencia cuadrada** | | |
|---|---|---|---|---|---|
| | Observada ($E$) | Esperada ($E$) | E-O | $(E-O)^2$ | Normalizada |
| 0.06 | 105 | 100 | -5 | 25 | 0.25 |
| 0.07 | 99 | 100 | 1 | 1 | 0.01 |
| 0.08 | 113 | 100 | -13 | 169 | 1.69 |
| 0.09 | 124 | 100 | -24 | 576 | 5.76 |
| 0.10 | 110 | 100 | -10 | 100 | 1 |
| 0.15 | 90 | 100 | 10 | 100 | 1 |
| 0.20 | 106 | 100 | -6 | 36 | 0.36 |
| 0.25 | 84 | 100 | 16 | 256 | 2.56 |
| 0.30 | 82 | 100 | 18 | 324 | 3.24 |
| 0.35 | 87 | 100 | 13 | 169 | 1.69 |
| **Total** | **1,000** | **1,000** | **0** | **1,756** | **17.56** |

un valor alfa ($\alpha$) para construir la solución. Para la afinación del parámetro $\alpha$ se decidió utilizar una estrategia reactiva en la cual el valor de $\alpha$ se adapta dinámicamente según los resultados obtenidos en las iteraciones previas, por lo tanto, se utiliza un conjunto discreto de valores predeterminados de alfas $A = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$.

En la experimentación llevada a cabo para la tesis, se tiene en total de $k = 10$ valores de alfa diferentes y se ejecutaron $n = 1,000$ iteraciones del algoritmo para cada instancia, actualizando la probabilidad de selección para cada alfa cada 100 iteraciones. Para la aplicación de la prueba $\chi^2$ se plantea como hipótesis nula ($H_0$) que las alfas en cada instancia son seleccionadas con la misma probabilidad y, como hipótesis alternativa ($H_1$) que las alfas en cada instancia no son seleccionadas con la misma probabilidad.

En el cuadro 1, se muestra la tabla de contingencia para la aplicación de la prueba $\alpha$. La frecuencia de cada alfa se tomó de los resultados arrojados por el algoritmo y el valor esperado de cada alfa se determinó mediante la ecuación 4:

$$\text{Frecuencia esperada} = P(\alpha) \times \text{n} \tag{4}$$

$$\text{Frecuencia esperada} = \frac{1}{10} \times 1000 \tag{5}$$

$$\text{Frecuencia esperada} = 100.$$

Para ejecutar la prueba de $\chi^2$ se utilizó la función `chisq.test()` del programa R, obteniendo como resultado un $\chi^2 \approx 17.56$ con 9 grados de libertad y un valor $p = 0.040$, por lo cual se procede a *rechazar* la hipótesis nula con un nivel de confianza del 95 %, lo que implica que al parecer se está seleccionando con mayor frecuencia algún valor de alfa dentro del conjunto dado. Este resultado tiene mucha lógica, ya que el algoritmo va seleccionando con mayor frecuencia aquellos valores de alfa que presentan mejor desempeño. El código empleado en el software R versión 4.0.2 para la prueba de chi cuadrada ($\chi^2$) se encuentra en el repositorio GitHub [1].

`test.txt`

```
        Chi-squared test for given probabilities

data:  tabla
X-squared = 17.56, df = 9, p-value = 0.04064
```

## 3. Demostraciones numérica y analítica

Sea $X$, $Y$ variables aleatorias y $a$, $b$, $c$, $d$ son constantes conocidas, demostrar que:

(a) $\text{Cov}(aX + b,\, cY + d) = ac\text{Cov}[(X,\, Y]$.

(b) $\text{Var}[X + Y] = \text{Var}[X] + \text{Var}[Y] + 2\text{Cov}[X, Y]$.

Con base en el desarrollo del teorema 6.2 y 6.4 del libro de Grinstead [2], se tiene que:

$$\begin{aligned}
\text{Cov}(aX + b,\, cY + d) &= E[(aX + b)(cY + d)] - E[(aX + b)]E[(cY + d)] \\
&= E(acXY + aXd + bcY + bd)[aE(X) + b][cE(Y) + d] \\
&= acE(XY) + adE(X) + bcE(Y) + bd - [acE(X)E(Y) + adE(X) + bcE(Y) + bd] \\
&= acE(XY) + adE(X) + bcE(Y) + bd - acE(X)E(Y) - adE(X) - bcE(Y) - bd \\
&= acE(XY)acE(X)E(Y) \\
&= ac\text{Cov}(X, Y). \qquad (6)
\end{aligned}$$

De acuerdo al desarrollo del teorema 6.6 y 6.8 del libro de Grinstead [2], se tiene que:

$$\begin{aligned}
\text{Var}[X+Y] &= E[(X+Y)^2][E(X)+E(Y)]^2 \\
&= E(X^2+2XY+Y^2) - [E(X)^2 + 2E(X)E(Y) + E(Y)^2] \\
&= E(X^2) + 2E(XY) + E(Y^2) - E(X)^2 - 2E(X)E(Y) - E(Y)^2 \\
&= E(X^2) - E(X)^2 + E(Y^2) - E(Y)^2 + 2[E(XY) - E(X)E(Y) \\
&= \text{Var}[X] + \text{Var}[Y] + 2\text{Cov}[X,Y]. \qquad (7)
\end{aligned}$$

Para la demostración numérica, se generaron 100 valores aleatorios para $X$ y $Y$ (donde no necesariamente son independientes) y se estableció $a = 5$, $b = 100$, $c = 20.5$ y $d = 0.85$. Se realizaron 100 replicas aplicando las ecuaciones 6 y 7, donde $X$ y $Y$ tenían diferente tipo de distribución (uniforme, normal, exponencial y Poisson). Los resultados de la simulación para la covarianza y varianza de estos valores se muestran en las figuras 3 y 4, respectivamente, en las que se puede observar que se cumple la igualdad de las ecuaciones 6 y 7, respectivamente, sin importar el tipo de distribución y o la independencia de variables aleatorias $X$ y $Y$ (ver figuras 3c y 4c).

Todas las simulaciones fueron realizadas en el software R versión 4.0.2 y el código empleado se encuentra en el repositorio GitHub [1].

# Referencias


[1] Bolaños Z., Johanna. Repositorio en GitHub de la clase de modelos probabilistas aplicados. Recursos libre, disponible en github.com/JohannaBZ/Probabilidad/tree/master/Tarea11, 2020.

[2] Grinstead, Charles M., Snell, J. Laurie. *Introduction to Probability.* American Mathematical Society, 2006.

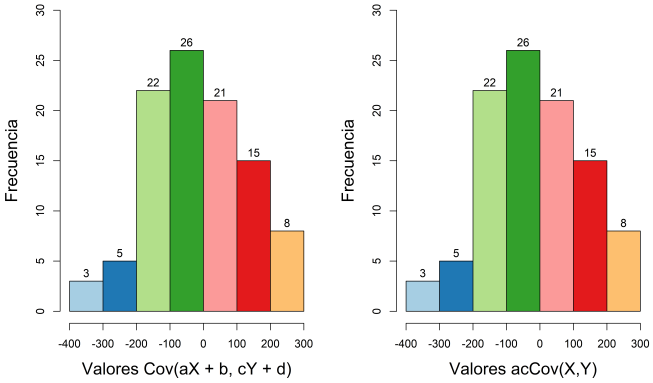
[3] Kim, S., Casper, R. Applications of Convolution in Image Processing with MATLAB. *University of Washington*, 2013.

[4] Mac Developer Library. Performing Convolution Operations. Recurso disponible en, https://developer.apple.com/library/archive/documentation/Performance/Conceptual/vImage/ConvolutionOpera 2016.

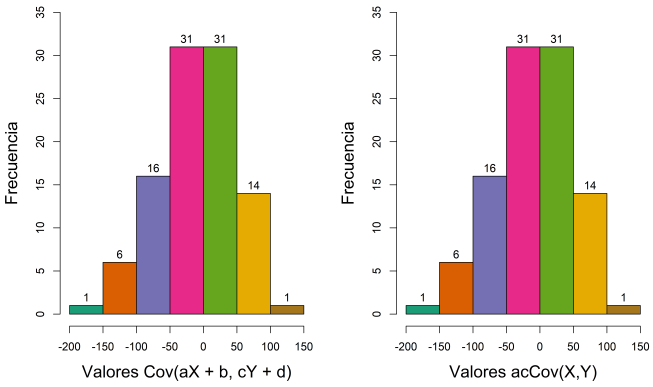
[5] Schaeffer, Elisa. Modelos probabilistas aplicados: notas del curso. Recurso disponible en, https://elisa.dyndns-web.com/teaching/prob/pisis/prob.htmlut2.

[6] The R Foundation. The R Project for Statistical Computing. https://www.r-project.org/, 2020.

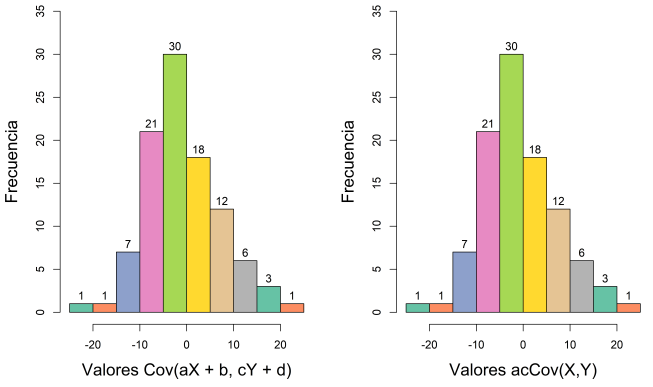
[7] Thyssen, Anthony. ImageMagick v6 Examples – Convolution of Images. Recurso disponible en, https://legacy.imagemagick.org/Usage/convolve/, 2013.




(a) Con $X \sim N(\mu, \sigma)$ y $Y \sim Pois(\lambda)$

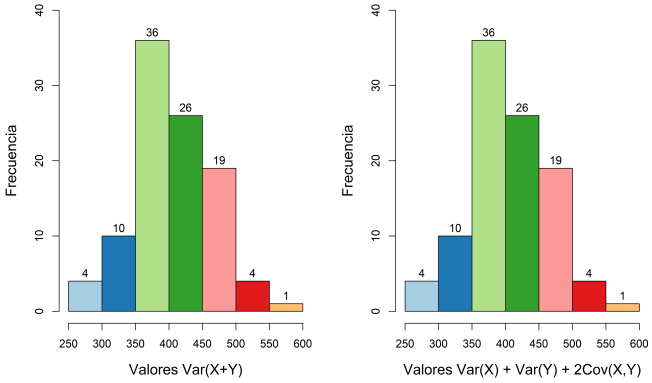



(b) Con $X \sim U(0,1)$ y $Y \sim Exp(\lambda)$

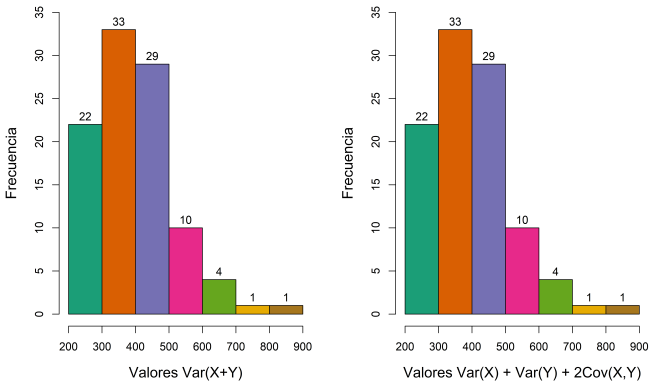



(c) Con $X \sim Exp(\lambda)$ y $Y \sim Pois(\lambda) + X$

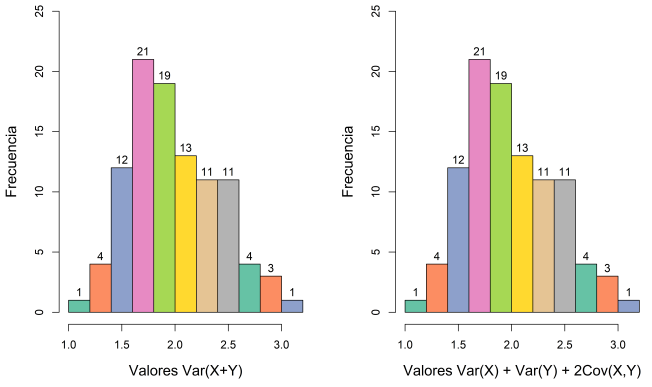
Figura 3: Resultados simulación para la covarianza




(a) Con $X \sim N(\mu, \sigma)$ y $Y \sim Pois(\lambda)$




(b) Con $X \sim U(0,1)$ y $Y \sim Exp(\lambda)$




(c) Con $X \sim Exp(\lambda)$ y $Y \sim Pois(\lambda) + X$

Figura 4: Resultados simulación para la varianza